SYSTEM CALL จากภาษา C

สมนึก ศรีวิศาล พ.บ.*

หัวข้อเรื่องก็บอกแล้วว่าจะมาพูดถึงภาษา C กัน เคยมีผู้กล่าวไว้ว่า ปี ค.ศ. 70 เป็นยุคของ CP/M ปี 80 ก็ต้องเป็นยุคของ C เมื่อเราอยู่ในทศวรรษนี้ จึงน่าจะต้องสนใจ C กันหน่อย และคงจะจริงอย่างที่เขาว่า เพราะตอนนี้ C คอมพายเลอร์ออกมามากมายเหลือเกิน ส่วนใหญ่อย่างที่ทราบกันแล้วว่า ใครๆก็เขียนมาเพื่อใช้กับไอบีเอ็มพีซี หรือเครื่องคอมแพตทั้งหลาย ราคานั้นมีต่างๆกัน เริ่มจากราว 50 เหรียญสหรัฐจนถึง 500 เหรียญก็มี คอมพายเลอร์จากแต่ละบริษัทจะให้ฟังก์ชันต่างๆมาไม่เท่ากัน แต่อย่างน้อยที่สุดคอมพายเลอร์จะมีฟังก์ชันพื้นฐานที่ปรากฏอยู่ในคัมภีร์ของภาษา C (C Programing Language โดย Kerighan & Ritchie) ทั้งนี้เพราะภาษา C เป็นภาษาที่ค่อนข้างเล็ก มีคำสั่งในภาษาแท้ๆไม่เท่าไร ที่สำคัญคือไม่มี input และ output คำสั่งพวกนี้ต้องสร้างเอาจากคำสั่งพื้นฐาน จะเห็นว่าคอมพายเลอร์ที่นิยมกันมาก เช่น Microsoft C หรือ Lattice C จะไม่มีฟังก์ชันบางอันที่ใช้กันบ่อยๆ ยกตัวอย่าง เช่น คำสั่งล้างจอ หรือคำสั่งที่วางตำแหน่งเคอร์เซอร์

จุดประสงค์ของบทความนี้ก็เพื่อจะแสดงให้เห็นว่าเราสามารถเขียนฟังก์ชันซึ่งไม่ได้ให้มาในคอมพายเลอร์ได้อย่างไร การเขียนฟังก์ชันพวกนี้ต้องอาศัย System Call เราก็จะพูดถึง System Call ด้วยว่าคืออะไรและจะใช้อย่างไร

ก่อนอื่นต้องบอกกล่าวกันก่อนว่าเครื่องที่พูดถึงในบทความนี้เป็นไอบีเอ็มพีซีหรือเครื่องคอมแพตเท่านั้น Operating System ที่กล่าวถึงคือ MS DOS โปรแกรมที่เขียนควบคู่มากับบทความนี้ได้ลองใช้กับเครื่องไอบีเอ็มเอ็กซ์ทีคอมแพตของผู้เขียน (Zenith PC) รวมทั้งเครื่องเลียนแบบที่ต่อใช้เอง (ซื้อชิ้นส่วนและบอร์ดต่างๆที่ทำจากไต้หวัน) ทุกเครื่องทำงานให้ผลเหมือนกันหมด DOS ที่ใช้เป็น MS DOS Version 2.11

คอมพายเลอร์ที่ผู้เขียนใช้เป็นของ Microsoft C Compiler Version 2.03 ซึ่งเหมือนกับ Lattice C Version 2.04 (ทั้งนี้เพราะ Microsoft ซื้อลิขสิทธิ์มาจากบริษัท Lifeboat) แม้กระทั่ง Microsoft C Compiler Version ล่าสุดคือ 3.00 ก็ยังไม่มีฟังก์ชันที่เอ่ยถึงข้างต้น คือ "clear screen" และ "position cursor"

ก่อนที่จะพูดกันถึงฟังก์ชันที่จะ Call Operating System เราต้องเข้าใจก่อนว่าพื้นฐานเป็นอย่างไร จะพูดถึงคร่าวๆพอให้เข้าใจเท่านั้น ในคอมพิวเตอร์ทุกเครื่องจะมีโปรแกรมซึ่งมากับเครื่องโดยบรรจุอยู่ในหน่วยความจำถาวร (ROM) โปรแกรมเหล่านี้จะทำหน้าที่พื้นฐานต่างๆ เช่น แสดงตัวหนังสือบนจอ CRT, ลบข้อความบนจอ หรือเอาเคอร์เซอร์ไปวางยังตำแหน่งต่างๆ หรือแม้จะเปลี่ยนรูปร่างของเคอร์เซอร์ เช่นจากสี่เหลี่ยมให้เป็นเส้น หรือจากสี่เหลี่ยมกะพริบให้เป็นไม่กะพริบ เป็นต้น หน้าที่ต่างๆนี้ทำงานโดยโปรแกรมดังกล่าว โปรแกรมที่ว่านี้ก็คือรอมไบออส (ROM BIOS) มีโปรแกรมอีกส่วนหนึ่งที่ทำหน้าที่ให้ความสะดวกแก่ผู้ใช้คือ Operating System ในที่นี้คือ MS DOS ซึ่งจะถูกบรรจุเข้าไปในเครื่องเวลาที่เราเปิดเครื่อง (boot up system) โปรแกรมส่วนนี้ทำหน้าที่ติดต่อกับดิสค์ไดรฟ์ เช่นจะดูว่าในแผ่นดิสค์มีโปรแกรมอะไรอยู่บ้าง ก็อาศัยคำสั่ง DIR สั่งผ่านดอสให้ทำงานให้เรา หน้าที่อื่นๆในรอมไบออสก็มีอีก เช่น ส่งตัวหนังสือออกไปพิมพ์ยังพรินเตอร์

ผู้ที่สนใจจะดูโปรแกรมในรอมไบออสได้ เพราะมีพิมพ์ใน IBM Technical Reference Manual โปรแกรมพวกนี้เขียนโดยใช้ภาษาแอสเซมบลีของ 8088/8086 ในเมื่อเรารู้ว่ามีโปรแกรมพวกนี้อยู่ที่ไหน ทำอะไรบ้าง เราก็สามารถดึงเอาส่วนของพวกนี้มาใช้ในโปรแกรมของเราได้ การที่ดึงเอาโปรแกรมพวกนี้มาใช้เรียกว่าเราทำ System Call เรา

*รังสีแพทย์ ST.LOUIS, MISSOURI.

สามารถดึงเอามาใช้โดยอาศัยภาษาอะไรก็ตามที่เรากำลังเขียนอยู่ได้ ถ้าหากเรารู้วิธีที่จะติดต่อกับรอมไบออสหรือดอส บทความนี้จะยกตัวอย่างวิธีทำ System Call จากภาษา C

เมื่อเราพูดถึงไอบีเอ็มพีซี เราก็ต้องรู้เกี่ยวกับตัวไมโครโปรเซสเซอร์ที่ใช้ในเครื่องสักเล็กน้อย เพื่อจะได้เข้าใจว่าพูดถึงอะไรกัน ไมโครโปรเซสเซอร์ที่ใช้เป็นของบริษัท Intel หมายเลข 8088/8086 ใน chip ตัวนี้ มีหน่วยทำงานที่เราเรียกว่า รีจิสเตอร์ (register คือที่เก็บข้อมูลใช้ในการทำงานต่างๆ เช่น ส่งข้อมูลไปเก็บในหน่วยความจำ, ทำหน้าที่บวก, ลบ, คูณ, หาร เป็นต้น)

รีจิสเตอร์ของ 8088/8086 มีชื่อต่างๆดังนี้ AX, BX, CX, DX, S1, D1, BP, SP, SS, ES, DS, & CS register เหล่านี้มีขนาด 16 bits นอกจากนี้ AX, BX, CX, DX ยังสามารถแตกออกเป็นสองส่วน แต่ละส่วนมีขนาดเล็กลงครึ่งหนึ่งของเดิม เช่น AX แยกเป็น AH, AL ทั้ง AH และ AL มีขนาด 8 บิท เป็นต้น ยกตัวอย่างว่า AX Register มีค่า 10FFH อยู่

เราจะเขียน AX = 10FFH
หรือจะเขียนในรูป AL = FFH
AH = 10H
จะมีค่าเท่ากัน

เวลาจะทำ System Call เราอาศัยวิธีการที่เรียกกันในภาษาแอสเซมบลีว่า interrupt call ซึ่งมีหมายเลขต่างๆกำกับแต่ละหมายเลขทำหน้าที่ต่างกัน เช่น ถ้าจะเรียก interrupt call หมายเลข 21H ก็หมายความว่าเราจะกลับไปยังดอส ถ้าเป็น interrupt call หมายเลข 10H ก็จะเกี่ยวกับการแสดงผลบนจอ การที่จะ call interrupt ทั้งหลายนี้ จะต้องเตรียมค่าในรีจิสเตอร์ต่างๆตามที่ interrupt ต้องการให้เรียบร้อยก่อนจะ call

ลองยกตัวอย่างสักอันหนึ่ง เช่น ถ้าเราจะล้างจอเราก็ต้องอาศัย interrupt ที่ทำหน้าที่เกี่ยวกับสัญญาณภาพ คือ interrupt 10 ซึ่งก่อนเรียกจะต้องเตรียมค่าของรีจิสเตอร์ต่างๆมีดังนี้

Register AL = 00H
AH = 06H

หรือเราจะรวม AL กับ AH ก็ได้ ถ้าจะเรียกรวมเป็น AX ก็จะได้ว่า

AX = 0600H
CX = 00H

(เป็นค่าของมุมซ้ายของบนจอ CRT)

DX = 184FH

(เป็นค่ามุมขวาล่างของจอ CRT)

BH = 7

(ค่าของ attribute)

เมื่อบรรจุค่าต่างๆตามที่ต้องการแล้ว เราก็ทำ interrupt call หมายเลข 10H ผลที่ได้ก็คือ ตัวหนังสือต่างๆบนจอจะหายไปหมด ตัวเคอร์เซอร์จะอยู่ที่มุมซ้ายบน (เหมือนคำสั่ง HOME ในภาษา BASIC APPLE SOFT) บางคนอาจจะสงสัยว่า เราจะทราบได้อย่างไรว่า register ไหนควรมีค่าเป็นเท่าใดก่อนทำ interrupt call อันนี้ต้องดูใน Technical Reference Manual ถ้าจะเขียนให้ครบเครื่องอย่างในเวลาเขียนภาษาแอสเซมบลีจะเขียนได้ดังนี้

CLS :
MOV AX, 0600H
MOV CX, 00H
MOV DX, 184FH
MOV BH, 07H
INT 10H

ทีนี้เรามาดูจากภาษา C ว่าเราจะติดต่อกับ System ได้อย่างไร ในคอมพายเลอร์ที่มีขายส่วนใหญ่จะมีฟังก์ชันที่เราจะใช้เรียกเพื่อให้ติดต่อกับ System ได้ ใน Microsoft C ฟังก์ชันนี้คือ

INT 86 (function NO, IN, OUT)

function NO คือหมายเลขฟังก์ชันใดที่เราจะเลือกใช้ เช่น 10H เป็นฟังก์ชันที่เกี่ยวกับวิดีโอ หรือหมายเลข 21H ก็คือฟังก์ชันที่ติดต่อกับดอส เป็นต้น

ต่อไปคือ IN คือค่าของรีจิสเตอร์ที่เราจะต้องกำหนดค่าเข้าไปก่อนที่จะ Call interrupt ดังตัวอย่างข้างต้น เราต้องใส่ค่า AX, CX, BH & DX เสียก่อน สรุปง่ายๆก็คือเราจะใช้ภาษาแอสเซมบลีโดยเรียกจากภาษา C สำหรับการเตรียมค่าต่างๆเพื่อผ่านไปยัง function call หรือเรียกตามภาษาคอมพิวเตอร์ว่า Parameter passing ใน System call เราใช้ Structure & Union ซึ่งเป็นคำสั่งในภาษา C (จะไม่กล่าวถึงรายละเอียดของ Structure และ Union เพราะเป็นเรื่องที่ตำราภาษา C ทั้งหลายพูดไว้เป็นบทใหญ่บทหนึ่ง ผู้สนใจจะหาอ่านได้จากตำราดังกล่าว) พูดย่อๆก็คือ Union เป็นที่ๆเรา pass parameter ไปให้ฟังก์ชัน

พารามิเตอร์ตัวสุดท้ายใน System Call คือ OUT อันนี้คือผลที่ได้มาจาก System call เวลาเรา call system บางหน้าที่ ยกตัวอย่างเช่น เราต้องการดูเวลาเราก็ Call DOS function โดยการใส่ค่า NO 21H เสร็จแล้ว set ค่า AH = GET__TIME (ในที่นี้เราตั้งค่า GET__TIME ไว้ตอนต้นของโปรแกรม) หลังจากที่ทำ System Call เราก็จะได้ค่าของเวลามาใน OUT ซึ่งเราจะเอาค่าใน OUT มาใช้พิมพ์บอกเวลา เป็นต้น

โปรแกรมตัวอย่างที่เขียนก็ตรงไปตรงมา สำคัญที่ต้องรู้ว่าคอมพายเลอร์ต้องการ set Structure และ Union ของรีจิสเตอร์ได้อย่างไร และต้องรู้ว่า System Call หมายเลขใด ต้องการค่าอะไรที่จะส่งให้บ้าง และจะได้ค่าอะไรกลับมาบ้าง

ส่วนแรกของโปรแกรมคือ main ( ) ซึ่งเป็นส่วนทดสอบฟังก์ชันต่างๆว่าทำหน้าที่ตามที่เราต้องการหรือเปล่า ที่เหลือก็เป็นฟังก์ชันต่างๆ เริ่มจาก CLS

( ) cursor (Row, Col) วางตำแหน่งของ Cursor ตามค่าพารามิเตอร์ที่เราต้องการคือ ตามแนวนอน และแถวเท่าไร function dostm ( ) เป็นตัวที่เรียกเวลาจากเครื่อง เหมือนกับเราพิมพ์ time เวลาที่อยู่ใน DOS environment บอกวันที่ให้เรา

โปรแกรม main ( ) เริ่มด้วยล้างจอก่อนเสร็จแล้วก็วางเคอร์เซอร์ที่บรรทัด 12 ตำแหน่ง 1 พิมพ์ข้อความต่อจากนั้นก็วางตำแหน่งเคอร์เซอร์บรรทัด 20 พิมพ์วันที่ที่ตำแหน่ง 41 เสร็จแล้วเราก็พิมพ์เวลาที่บรรทัด 22 ตำแหน่ง 41 เช่นกันคือใต้วันที่

ผู้เขียนหวังว่า บทความนี้คงจะเป็นประโยชน์แก่ผู้เริ่มเรียน C พอควร โดยเฉพาะ CLS ( ) และ Cursor ( ) ค่อนข้างจะใช้มากในโปรแกรมที่ต้องการจะ format screen output

โปรแกรมตัวอย่าง

```
/*      program title   func.c

*       function names: main(),cls(),cursor(),dosdt(),dostm()
*
*       program craftsman : Somnuk Srivisal
*
*       compile with : stdio.h
*
*       compiler :MICROSOFT C Version 2.03
*
*       description :
*                       The main() function is for demonstration
*       of the function calls.
*                       The Microsoft C compiler version 2.03 is
*       the same as Lattice C compiler 2.04. Mark Williams C compiler
*       (Let C) and the above mentioned compilers do not provide clear
*       screen and cursor positioning functions.
*                       This program shows how these 2 functions
*       are created.The cls() works the same way as cls in the BASIC
*       and dos environment.The cursor(row,col) works the same way as
*       gotoxy() in Turbo Pascal.
*                       Two more functios are also added.Using dos
*       call function 'get_time' and 'get_date'.Use dostm() to get
*       time and dosdt() to get date.
*/
#include        <stdio.h>

#define         DOSCALL         0x21 /* interupt H21 */
#define         VIDEO           0x10 /* interupt H10 */

#define         GET_DATE        0x2a
#define         GET_TIME        0x2c

struct          XREGS
                {
short           ax ,    /* 8088/8086 registers */
                bx ,    /* 16 bits */
                cx ,
                dx ,
                si ,
                di ;
                } ;

struct          HREGS
                {
char
                al ,    /* 8088/8086 registers */
                ah ,    /* 8 bits */
                bl ,
                bh ,
                cl ,
```

```
                ch ,
                dl ,
                dh ;
                } ;
union REGS
                {
                struct XREGS x ;
                struct HREGS h ;
                } ;

union   REGS    inregs,outregs ;

#define         CENTER  12

main()

{

        cls() ;         /* clear screen function */

        cursor(CENTER,0) ;      /* position cursor to line 12 col 0 */

        puts("Start printing on line 12 column 1") ;

        cursor(20,0);   /* position cursor to line 23 column 20 */

        puts("print time on line 20 column 40");

        cursor(20,40);

        dosdt();        /* print date at this location */

        dostm();        /* print time next line to date */

}

/*      ------------------------------
*
*       this function will clear screen
*       and home the cursor.No return value.
*
        ------------------------------ */

cls()
        {
        inregs.x.ax = 0x0600  ;/* set ax to h0600 */
                              /* <==> al=00,ah=06 */
        inregs.x.cx = 0x0   ;   /* upper left corner */
        inregs.x.dx = 0x184f  ; /* lower right corner */
        inregs.h.bh = 7   ;     /* set attribute */

        /* this is how we call interupt from C */

        int86(VIDEO,&inregs,&outregs) ;
        }

/*      ------------------------------
*
*       this function will set cursor to position
*       row and col.No return value.
*
        ------------------------------ */

cursor(row,col)
        int     row,col ;
        {
        inregs.h.ah = 2 ;       /* set cursor function */
        inregs.h.dh = row ;
```

```
        inregs.h.dl = col ;
        inregs.h.bh = 0 ;       /* set vedio page 0 */

        /* call interupt h10 */

        int86(VIDEO,&inregs,&outregs);
        }

/*      ------------------------------
*
*               This function will print date
*       in this dd/mm/yyyy format.
*
*               On return from interupt call H21 function H2a,
*
*          CX = year(1980-2099)
*          DH = month(1-12)
*          DL = day(1-31)
*
*       are stored in the outregs.
*               Therefore we have to convert these values by using
*       printf() function.
*
*       ------------------------------ */

dosdt()
        {
        inregs.h.ah = GET_DATE; /* set ah = H2a) */

        /* call dos interupt (H21) */

        int86(DOSCALL,&inregs,&outregs);

        printf("date : %d/%d/%d\n",outregs.h.dl,outregs.h.dh,outregs.x.c
x);

        }

/*      ------------------------------
*
*               This function will print time
*       in this hh:min:sec.xx
*
*               On return from function call,
*
*               CH = hour(0-23)
*               CL = min.(0-59)
*               DH = sec.(0-59)
*               DL = 1/100(0-99)
*
*       are stored in outregs.
*               we have to convert by using
*       printf() function.
*
*       ------------------------------ */

dostm()
        {
        inregs.h.ah = GET_TIME ; /* set ah = H2c */

        /* call dos interupt (H21) */

        int86(DOSCALL,&inregs,&outregs);

        printf("\ntime = %d:%d:%d.%d\n",outregs.h.ch,outregs.h.cl,
                        outregs.h.dh,outregs.h.dl);
        }

/*****************************************************************/
```